Ensemble Methods

+ 

Recommender Systems
Reminders

• Homework 8: Reinforcement Learning
  – Out: Mon, Nov. 21
  – Due: Fri, Dec. 2 at 11:59pm

• Exam 2 Exit Poll
  – Due: Fri, Dec. 2 at 11:59pm

• Homework 9: Learning Paradigms
  – Out: Fri, Dec. 2
  – Due: Fri, Dec. 9 at 11:59pm
  (only two grace/late days permitted)
Learning Paradigms

<table>
<thead>
<tr>
<th>Paradigm</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised</td>
<td>$D = {x^{(i)}, y^{(i)}}_{i=1}^N \quad x \sim p^<em>(\cdot)$ and $y = c^</em>(\cdot)$</td>
</tr>
<tr>
<td>Regression</td>
<td>$y^{(i)} \in \mathbb{R}$</td>
</tr>
<tr>
<td>Classification</td>
<td>$y^{(i)} \in {1, \ldots, K}$</td>
</tr>
<tr>
<td>Binary classification</td>
<td>$y^{(i)} \in {+1, -1}$</td>
</tr>
<tr>
<td>Structured Prediction</td>
<td>$y^{(i)}$ is a vector</td>
</tr>
<tr>
<td>Unsupervised</td>
<td>$D = {x^{(i)}}_{i=1}^N \quad x \sim p^*(\cdot)$</td>
</tr>
<tr>
<td>Clustering</td>
<td>predict ${z^{(i)}}_{i=1}^N$ where $z^{(i)} \in {1, \ldots, K}$</td>
</tr>
<tr>
<td>Dimensionality Reduction</td>
<td>convert each $x^{(i)} \in \mathbb{R}^M$ to $u^{(i)} \in \mathbb{R}^K$ with $K \ll M$</td>
</tr>
<tr>
<td>Semi-supervised</td>
<td>$D = {x^{(i)}, y^{(i)}}<em>{i=1}^{N_1} \cup {x^{(j)}}</em>{j=1}^{N_2}$</td>
</tr>
<tr>
<td>Online</td>
<td>$D = {(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \ldots}$</td>
</tr>
<tr>
<td>Active Learning</td>
<td>$D = {x^{(i)}}_{i=1}^N$ and can query $y^{(i)} = c^*(\cdot)$ at a cost</td>
</tr>
<tr>
<td>Imitation Learning</td>
<td>$D = {(s^{(1)}, a^{(1)}), (s^{(2)}, a^{(2)}), \ldots}$</td>
</tr>
<tr>
<td>Reinforcement Learning</td>
<td>$D = {(s^{(1)}, a^{(1)}, r^{(1)}), (s^{(2)}, a^{(2)}, r^{(2)}), \ldots}$</td>
</tr>
</tbody>
</table>
**ML Big Picture**

### Learning Paradigms:
**What data is available and when? What form of prediction?**
- supervised learning
- unsupervised learning
- semi-supervised learning
- reinforcement learning
- active learning
- imitation learning
- domain adaptation
- online learning
- density estimation
- recommender systems
- feature learning
- manifold learning
- dimensionality reduction
- ensemble learning
- distant supervision
- hyperparameter optimization

### Problem Formulation:
**What is the structure of our output prediction?**
- boolean: Binary Classification
- categorical: Multiclass Classification
- ordinal: Ordinal Classification
- real: Regression
- ordering: Ranking
- multiple discrete: Structured Prediction
- multiple continuous: (e.g. dynamical systems)
- both discrete & cont.: (e.g. mixed graphical models)

### Facets of Building ML Systems:
**How to build systems that are robust, efficient, adaptive, effective?**
1. Data prep
2. Model selection
3. Training (optimization / search)
4. Hyperparameter tuning on validation data
5. (Blind) Assessment on test data

### Theoretical Foundations:
**What principles guide learning?**
- probabilistic
- information theoretic
- evolutionary search
- ML as optimization

### Application Areas
Key challenges: NLP, Speech, Computer Vision, Robotics, Medicine, Search

### Big Ideas in ML:
Which are the ideas driving development of the field?
- inductive bias
- generalization / overfitting
- bias-variance decomposition
- generative vs. discriminative
- deep nets, graphical models
- PAC learning
- distant rewards
Outline for Today

We’ll talk about two distinct topics:

1. **Ensemble Methods**: combine or learn multiple classifiers into one (i.e. a family of algorithms)

2. **Recommender Systems**: produce recommendations of what a user will like (i.e. the solution to a particular type of task)

We’ll use a prominent example of a recommender systems (the Netflix Prize) to motivate both topics...
RECOMMENDER SYSTEMS
A Common Challenge:

– Assume you’re a company selling **items** of some sort: movies, songs, products, etc.

– Company collects millions of **ratings** from **users** of their **items**

– To maximize profit / user happiness, you want to **recommend** items that users are likely to want
Recommender Systems
Recommender Systems

Congratulations!

The Netflix Prize sought to substantially improve the accuracy of predictions about how much someone is going to enjoy a movie based on their movie preferences.

On September 21, 2009 we awarded the $1M Grand Prize to team “BellKor’s Pragmatic Chaos”. Read about their algorithm, checkout team scores on the Leaderboard, and join the discussions on the Forum.

We applaud all the contributors to this quest, which improves our ability to connect people to the movies they love.
Recommender Systems

Netflix Prize

The Netflix Prize sought to substantially improve the accuracy of predictions about how much someone is going to enjoy a movie based on their movie preferences.

On September 21, 2009 we awarded the $1M Grand Prize to team "BellKor's Pragmatic Chaos". Read about their algorithm, checkout team scores on the Leaderboard, and join the discussions on the Forum.

We applaud all the contributors to this quest, which improves our ability to connect people to the movies they love.
Problem Setup

- 500,000 users
- 20,000 movies
- 100 million ratings
- Goal: To obtain lower root mean squared error (RMSE) than Netflix’s existing system on 3 million held out ratings
ENSEMBLE METHODS
Top performing systems were ensembles
Weighted Majority Algorithm

(Littlestone & Warmuth, 1994)

- **Given**: pool $A$ of binary classifiers (that you know nothing about)
- **Data**: stream of examples (i.e. online learning setting)
- **Goal**: design a new learner that uses the predictions of the pool to make new predictions
- **Algorithm**:
  - Initially weight all classifiers equally
  - Receive a training example and predict the (weighted) majority vote of the classifiers in the pool
  - Down-weight classifiers that contribute to a mistake by a factor of $\beta$
Weighted Majority Algorithm
(Littlestone & Warmuth, 1994)

Suppose we have a pool of $T$ binary classifiers $\mathcal{A} = \{h_1, \ldots, h_T\}$ where $h_t : \mathbb{R}^M \rightarrow \{+1, -1\}$. Let $\alpha_t$ be the weight for classifier $h_t$.

**Algorithm 1** Weighted Majority Algorithm

1: **procedure** WEIGHTEDMAJORITY($\mathcal{A}$, $\beta$)
2: Initialize classifier weights $\alpha_t = 1$, $\forall t \in \{1, \ldots, T\}$
3: **for** each training example $(x, y)$ **do**
4: Predict majority vote class (splitting ties randomly)

$$\hat{h}(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$$

5: **if** a mistake is made $\hat{h}(x) \neq y$ **then**
6: **for** each classifier $t \in \{1, \ldots, T\}$ **do**
7: If $h_t(x) \neq y$, then $\alpha_t \leftarrow \beta \alpha_t$
Weighted Majority Algorithm

**Theorems** (Littlestone & Warmuth, 1994)

For the general case where $WM$ is applied to a pool $\mathcal{A}$ of algorithms we show the following upper bounds on the number of mistakes made in a given sequence of trials:

1. $O(\log |\mathcal{A}| + m)$, if one algorithm of $\mathcal{A}$ makes at most $m$ mistakes.

2. $O(\log \frac{|\mathcal{A}|}{k} + m)$, if each of a subpool of $k$ algorithms of $\mathcal{A}$ makes at most $m$ mistakes.

3. $O(\log \frac{|\mathcal{A}|}{k} + \frac{m}{k})$, if the total number of mistakes of a subpool of $k$ algorithms of $\mathcal{A}$ is at most $m$.

These are “mistake bounds” of the variety we saw for the Perceptron algorithm.
ADABOOST
Comparison

**Weighted Majority Algorithm**
- an example of an ensemble method
- assumes the classifiers are learned ahead of time
- only learns (majority vote) weight for each classifiers

**AdaBoost**
- an example of a boosting method
- simultaneously learns:
  - the classifiers themselves
  - (majority vote) weight for each classifiers
weak classifiers = vertical or horizontal half-planes
AdaBoost: Toy Example

$h_1$

$\varepsilon_1 = 0.30$
$\alpha_1 = 0.42$

$D_2$
AdaBoost: Toy Example

\[ \alpha_2 = 0.65 \]
\[ \epsilon_2 = 0.21 \]
AdaBoost: Toy Example
$H_{\text{final}} = \text{sign}(0.42 + 0.65 + 0.92)$

Slide from Schapire NIPS Tutorial
AdaBoost

Given: \((x_1, y_1), \ldots, (x_m, y_m)\) where \(x_i \in X, y_i \in Y = \{-1, +1\}\)

Initialize \(D_1(i) = 1/m\).

For \(t = 1, \ldots, T\):

- Train weak learner using distribution \(D_t\).
- Get weak hypothesis \(h_t : X \to \{-1, +1\}\) with error
  \[
  \epsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i].
  \]
- Choose \(\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t}\right)\).
- Update:
  \[
  D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \left\{ \begin{array}{ll}
  e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\
  e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i
  \end{array} \right.
  \]
  \[
  = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}
  \]
  where \(Z_t\) is a normalization factor (chosen so that \(D_{t+1}\) will be a distribution).

Output the final hypothesis:

\[
H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right).
\]
Figure 2: Error curves and the margin distribution graph for boosting C4.5 on the letter dataset as reported by Schapire et al. [41]. Left: the training and test error curves (lower and upper curves, respectively) of the combined classifier as a function of the number of rounds of boosting. The horizontal lines indicate the test error rate of the base classifier as well as the test error of the final combined classifier. Right: The cumulative distribution of margins of the training examples after 5, 100 and 1000 iterations, indicated by short-dashed, long-dashed (mostly hidden) and solid curves, respectively.
Learning Objectives

Ensemble Methods / Boosting

You should be able to...

1. Implement the Weighted Majority Algorithm
2. Implement AdaBoost
3. Distinguish what is learned in the Weighted Majority Algorithm vs. Adaboost
4. Contrast the theoretical result for the Weighted Majority Algorithm to that of Perceptron
5. Explain a surprisingly common empirical result regarding Adaboost train/test curves
RECOMMENDER SYSTEMS
Problem Setup

- 500,000 users
- 20,000 movies
- 100 million ratings
- Goal: To obtain lower root mean squared error (RMSE) than Netflix’s existing system on 3 million held out ratings
Recommender Systems

![Image of Netflix Prize Leaderboard]

<table>
<thead>
<tr>
<th>Rank</th>
<th>Team Name</th>
<th>Best Test Score</th>
<th>% Improvement</th>
<th>Best Submit Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BellKor's Pragmatic Chaos</td>
<td>0.8567</td>
<td>10.06</td>
<td>2009-07-26 18:18:28</td>
</tr>
<tr>
<td>2</td>
<td>The Ensemble</td>
<td>0.8567</td>
<td>10.06</td>
<td>2009-07-26 18:38:22</td>
</tr>
<tr>
<td>3</td>
<td>Grand Prize Team</td>
<td>0.8582</td>
<td>9.90</td>
<td>2009-07-10 21:24:40</td>
</tr>
<tr>
<td>4</td>
<td>Opera Solutions and Vandelay United</td>
<td>0.8588</td>
<td>9.84</td>
<td>2009-07-10 01:12:31</td>
</tr>
<tr>
<td>5</td>
<td>Vandelay Industries 1</td>
<td>0.8591</td>
<td>9.81</td>
<td>2009-07-10 00:32:20</td>
</tr>
<tr>
<td>6</td>
<td>PragmaticTheory</td>
<td>0.8594</td>
<td>9.77</td>
<td>2009-06-24 12:06:56</td>
</tr>
<tr>
<td>7</td>
<td>BellKor in BigChaos</td>
<td>0.8601</td>
<td>9.70</td>
<td>2009-05-13 08:14:09</td>
</tr>
<tr>
<td>8</td>
<td>Dace</td>
<td>0.8612</td>
<td>9.59</td>
<td>2009-07-24 17:18:43</td>
</tr>
<tr>
<td>9</td>
<td>Feeds2</td>
<td>0.8622</td>
<td>9.48</td>
<td>2009-07-12 13:11:51</td>
</tr>
<tr>
<td>10</td>
<td>BigChaos</td>
<td>0.8623</td>
<td>9.47</td>
<td>2009-04-07 12:33:59</td>
</tr>
<tr>
<td>11</td>
<td>Opera Solutions</td>
<td>0.8623</td>
<td>9.47</td>
<td>2009-07-24 00:34:07</td>
</tr>
<tr>
<td>12</td>
<td>BellKor</td>
<td>0.8624</td>
<td>9.46</td>
<td>2009-07-26 17:19:11</td>
</tr>
</tbody>
</table>
Recommender Systems

• **Setup:**
  – **Items:** movies, songs, products, etc. (often many thousands)
  – **Users:** watchers, listeners, purchasers, etc. (often many millions)
  – **Feedback:** 5-star ratings, not-clicking ‘next’, purchases, etc.

• **Key Assumptions:**
  – Can represent ratings numerically as a user/item matrix
  – Users only rate a small number of items (the matrix is sparse)

<table>
<thead>
<tr>
<th></th>
<th>Doctor</th>
<th>Strange</th>
<th>Star Trek: Beyond</th>
<th>Zootopia</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Charlie</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Two Types of Recommender Systems

Content Filtering
• Example: Pandora.com music recommendations (Music Genome Project)
• Con: Assumes access to side information about items (e.g. properties of a song)
• Pro: Got a new item to add? No problem, just be sure to include the side information

Collaborative Filtering
• Example: Netflix movie recommendations
• Pro: Does not assume access to side information about items (e.g. does not need to know about movie genres)
• Con: Does not work on new items that have no ratings
COLLABORATIVE FILTERING
Collaborative Filtering

• Everyday Examples of Collaborative Filtering...
  – Bestseller lists
  – Top 40 music lists
  – The “recent returns” shelf at the library
  – Unmarked but well-used paths thru the woods
  – The printer room at work
  – “Read any good books lately?”
  – ...

• **Common insight:** personal tastes are correlated
  – If Alice and Bob both like X and Alice likes Y then Bob is more likely to like Y
  – especially (perhaps) if Bob knows Alice
Two Types of Collaborative Filtering

1. Neighborhood Methods

2. Latent Factor Methods

Figures from Koren et al. (2009)
Two Types of Collaborative Filtering

1. Neighborhood Methods

Algorithm:
1. Find neighbors based on similarity of movie preferences
2. Recommend movies that those neighbors watched

In the figure, assume that a green line indicates the movie was watched.

Figures from Koren et al. (2009)
Two Types of Collaborative Filtering

2. Latent Factor Methods

• Assume that both movies and users live in some low-dimensional space describing their properties

• **Recommend** a movie based on its **proximity** to the user in the latent space

• **Example Algorithm:** Matrix Factorization

Figures from Koren et al. (2009)
Question:
Applied to the Netflix Prize problem, which of the following methods *always* requires side information about the users and movies?

**Select all that apply**
A. K-Means
B. collaborative filtering
C. latent factor methods
D. ensemble methods
E. content filtering
F. neighborhood methods
G. recommender systems

Answer:
MATRIX FACTORIZATION
Matrix Factorization

• Many different ways of factorizing a matrix
• We’ll consider three:
  1. Unconstrained Matrix Factorization
  2. Singular Value Decomposition
  3. Non-negative Matrix Factorization

• MF is just another example of a common recipe:
  1. define a model
  2. define an objective function
  3. optimize with SGD
Matrix Factorization

Whiteboard

– Background: Low-rank Factorizations
– Residual matrix
Example: MF for Netflix Problem

(a) Example of rank-2 matrix factorization

(b) Residual matrix

Figures from Aggarwal (2016)
Regression vs. Collaborative Filtering

Figure 3.1: Revisiting Figure 1.4 of Chapter 1. Comparing the traditional classification problem with collaborative filtering. Shaded entries are missing and need to be predicted.

The value of the missing entries need to be learned for the test data. This scenario is illustrated in Figure 3.1(a), where the shaded values represent missing entries in the matrix.

Unlike data classification, any entry in the ratings matrix may be missing, as illustrated by the shaded entries in Figure 3.1(b). Thus, it can be clearly seen that the matrix completion problem is a generalization of the classification (or regression modeling) problem.

Therefore, the crucial differences between these two problems may be summarized as follows:

1. In the data classification problem, there is a clear separation between feature (independent) variables and class (dependent) variables. In the matrix completion problem, this clear separation does not exist. Each column is both a dependent and independent variable, depending on which entries are being considered for predictive modeling at a given point.

2. In the data classification problem, there is a clear separation between the training and test data. In the matrix completion problem, this clear demarcation does not exist among the rows of the matrix. At best, one can consider the specified (observed) entries to be the training data, and the unspecified (missing) entries to be the test data.

3. In data classification, columns represent features, and rows represent data instances. However, in collaborative filtering, it is possible to apply the same approach to either the ratings matrix or to its transpose because of how the missing entries are distributed. For example, user-based neighborhood models can be viewed as direct...

Figures from Aggarwal (2016)
UNCONSTRAINED MATRIX FACTORIZATION
Unconstrained Matrix Factorization

Whiteboard

– Optimization problem
– SGD
– SGD with Regularization
– Alternating Least Squares
– User/item bias terms (matrix trick)
Unconstrained Matrix Factorization

SGD for UMF:

\[
\text{While not converged:} \\
\text{(1) Sample } (i, j) \text{ from } Z \text{ uniformly at random} \\
\text{(2) Compute } e_{ij} = r_{ij} - \hat{u}_i^T \hat{v}_j \\
\text{(3) Update} \\
\hat{u}_i \leftarrow \hat{u}_i - \gamma \nabla_{\hat{u}_i} J_{ij}(U,V) \\
\hat{v}_j \leftarrow \hat{v}_j - \gamma \nabla_{\hat{v}_j} J_{ij}(U,V) \\
\text{where } e_{ij} = r_{ij} - \hat{u}_i^T \hat{v}_j
\]

\[
J_{ij}(U,V) = \frac{1}{2} (r_{ij} - \hat{u}_i^T \hat{v}_j)^2 + \lambda (\|u_i\|_2^2 + \|v_j\|_2^2)
\]

\[
\nabla_{\hat{u}_i} J_{ij}(U,V) = -e_{ij} \hat{v}_j + \lambda \hat{u}_i \\
\nabla_{\hat{v}_j} J_{ij}(U,V) = -e_{ij} \hat{u}_i + \lambda \hat{v}_j
\]
Unconstrained Matrix Factorization

SGD for UMF:

\[ r_{ij} = o_i + p_j + \hat{U}_i \hat{V}_j \]

\[ U = \begin{bmatrix} u_1 & u_2 & \cdots & u_m \end{bmatrix} \]

\[ V = \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix} \]
Unconstrained Matrix Factorization

Alternating Least Squares (ALS) for UMF:

Block Coord. Descent:

While not converged:
1. $U = \text{argmin}_U J(U,V)$
2. $V = \text{argmin}_V J(U,V)$

$J(U,V) = \frac{1}{2} \sum_{(i,j) \in E} (r_{ij} - u_i^T v_j)^2$

if $U$ is fixed
Least Squares in $V$

if $V$ is fixed
Least Squares in $U$

Option 1: take derivatives, set to zero and solve in closed form

* solving $J(U,V)$ in closed form directly isn’t easy
and $J(U,V)$ is non-convex
Figure 3. The first two vectors from a matrix decomposition of the Netflix Prize data. Selected movies are placed at the appropriate spot based on their factor vectors in two dimensions. The plot reveals distinct genres, including clusters of movies with strong female leads, fraternity humor, and quirky independent films.
Matrix Factorization

Comparison of Optimization Algorithms

ALS = alternating least squares

Figure from Gemulla et al. (2011)
SVD FOR COLLABORATIVE FILTERING
Singular Value Decomposition for Collaborative Filtering

For any arbitrary matrix $A$, SVD gives a decomposition:

$$A = U\Lambda V^T$$

where $\Lambda$ is a diagonal matrix, and $U$ and $V$ are orthogonal matrices.

Suppose we have the SVD of our ratings matrix

$$R = Q\Sigma P^T,$$

but then we truncate each of $Q$, $\Sigma$, and $P$ s.t. $Q$ and $P$ have only $k$ columns and $\Sigma$ is $k \times k$:

$$R \approx Q_k \Sigma_k P_k^T$$

For collaborative filtering, let:

$$U \triangleq Q_k \Sigma_k$$

$$V \triangleq P_k$$

$$\Rightarrow U, V = \text{argmin}_{U,V} \frac{1}{2} \| R - UV^T \|_2^2$$

s.t. columns of $U$ are mutually orthogonal

s.t. columns of $V$ are mutually orthogonal

**Theorem:** If $R$ fully observed and no regularization, the optimal $UV^T$ from SVD equals the optimal $UV^T$ from Unconstrained MF
NON-NEGATIVE MATRIX FACTORIZATION
Implicit Feedback Datasets

• What information does a five-star rating contain?

• Implicit Feedback Datasets:
  – In many settings, users don’t have a way of expressing dislike for an item (e.g. can’t provide negative ratings)
  – The only mechanism for feedback is to “like” something

• Examples:
  – Facebook has a “Like” button, but no “Dislike” button
  – Google’s “+1” button
  – Pinterest pins
  – Purchasing an item on Amazon indicates a preference for it, but there are many reasons you might not purchase an item (besides dislike)
  – Search engines collect click data but don’t have a clear mechanism for observing dislike of a webpage

Examples from Aggarwal (2016)
Non-negative Matrix Factorization

Constrained Optimization Problem:

\[ U, V = \arg\min_{U,V} \frac{1}{2} \| R - UV^T \|_2^2 \]

s.t. \( U_{ij} \geq 0 \)

s.t. \( V_{ij} \geq 0 \)

**Multiplicative Updates**: simple iterative algorithm for solving just involves multiplying a few entries together
Fighting Fire with Fire: Using Antidote Data to Improve Polarization and Fairness of Recommender Systems

Bashir Rastegarpanah
Boston University
bashir@bu.edu

Krishna P. Gummadli
MPI-SWS
gummadi@mpi-sws.org

Mark Crovella
Boston University
crovella@bu.edu

5 SOCIAL OBJECTIVE FUNCTIONS

The previous section developed a general framework for improving various properties of recommender systems; in this section we show how to apply that framework specifically to issues of polarization and fairness.

As described in Section 2, polarization is the degree to which opinions, views, and sentiments diverge within a population. Recommender systems can capture this effect through the ratings that they present for items. To formalize this notion, we define polarization in terms of the variability of predicted ratings when compared across users. In fact, we note that both very high variability, and very low variability of ratings may be undesirable. In the case of high variability, users have strongly divergent opinions, leading to conflict. Recent analyses of the YouTube recommendation system have suggested that it can enhance this effect [29, 30]. On the other hand, the convergence of user preferences, i.e., very low variability of ratings given to each item across users, corresponds to increased homogeneity, an undesirable phenomenon that may occur as users interact with a recommender system [11]. As a result, in what follows we consider using antidote data in both ways to either increase or decrease polarization.

As also described in Section 2, unfairness is a topic of growing interest in machine learning. Following the discussion in that section, we consider a recommender system fair if it provides equal quality of service (i.e., prediction accuracy) to all users or all groups of users [36].

Next we formally define the metrics that specify the objective functions associated with each of the above objectives. Since the gradient of each objective function is used in the optimization algorithm, for reproducibility we provide the details about derivation of the gradients in appendix A.2.

5.1 Polarization

To capture polarization, we seek to measure the extent to which the user ratings disagree. Thus, to measure user polarization we consider the estimated ratings \( \hat{X} \), and we define the polarization metric as the normalized sum of pairwise euclidean distances between estimated user ratings, i.e., between rows of \( \hat{X} \). In particular:

\[
R_{pol}(\hat{X}) = \frac{1}{n^2} \sum_{i \neq j} \| \hat{x}_i - \hat{x}_j \|^2
\]  

(10)

The normalization term \( \frac{1}{n^2} \) in (10) makes the polarization metric identical to the following definition:

\[
R_{pol} = \frac{1}{d} \sum_{j=1}^{d} a_j^2
\]  

(11)

where \( a_j^2 \) is the variance of estimated user ratings for item \( j \). Thus this polarization metric can be interpreted either as the average of the variances of estimated ratings in each item, or equivalently as the average user disagreement over all items.

5.2 Fairness

Individual fairness. For each user \( i \), we define \( \ell_i \), the loss of user \( i \), as the mean squared estimation error over known ratings of user \( i \):

\[
\ell_i = \frac{\| P_{i} (\hat{X} - X)^2 \|}{\| P_{i} \|}
\]  

(12)

Then we define the individual unfairness as the variance of the user losses:

\[
R_{indiv}(X, \hat{X}) = \frac{1}{n^2} \sum_{i \neq j} (\ell_i - \ell_j)^2
\]  

(13)

To improve individual fairness, we seek to minimize \( R_{indiv} \).

Group fairness. Let \( L \) be the set of all users/items and \( G = \{ G_1, \ldots, G_g \} \) be a partition of users/items into \( g \) groups, i.e., \( L = \bigcup_{i=1}^{g} G_i \). We define the loss of group \( i \) as the mean squared estimation error over all known ratings in group \( i \):

\[
L_i = \frac{\| P_{G_i} (\hat{X} - X)^2 \|}{\| P_{G_i} \|}
\]  

(14)

For a given partition \( G \), we define the group unfairness as the variance of all group losses:

\[
R_{grp}(X, \hat{X}, G) = \frac{1}{g^2} \sum_{i \neq j} (L_i - L_j)^2
\]  

(15)

Again, to improve group fairness, we seek to minimize \( R_{grp} \).

5.3 Accuracy vs. Social Welfare

Adding antidote data to the system to improve a social utility will also have an effect on the overall prediction accuracy. Previous works have considered social objectives as regularizers or constraints added to the recommender model (e.g., [8, 25, 37]), implying a trade-off between the prediction accuracy and a social objective.

However, in the case of the metrics we define here, the relationship is not as simple. Considering polarization, we find that in general, increasing or decreasing polarization will tend to decrease system accuracy. In either case we find that system accuracy only declines slightly in our experiments, we report on the specific values in Section 6. Considering either individual or group unfairness, the situation is more subtle. Note that our unfairness metrics will be exactly zero for a system with zero error (perfect accuracy). As a
Summary

• Recommender systems solve many real-world (*large-scale) problems
• Collaborative filtering by Matrix Factorization (MF) is an efficient and effective approach
• MF is just another example of a common recipe:
  1. define a model
  2. define an objective function
  3. optimize with your favorite black box optimizer (e.g. SGD, Gradient Descent, Block Coordinate Descent aka. Alternating Least Squares)
Learning Objectives

Recommender Systems

You should be able to...

1. Compare and contrast the properties of various families of recommender system algorithms: content filtering, collaborative filtering, neighborhood methods, latent factor methods

2. Formulate a squared error objective function for the matrix factorization problem

3. Implement unconstrained matrix factorization with a variety of different optimization techniques: gradient descent, stochastic gradient descent, alternating least squares

4. Offer intuitions for why the parameters learned by matrix factorization can be understood as user factors and item factors
EXTRA SLIDES ON UMF
In-Class Exercise

Derive a block coordinate descent algorithm for the Unconstrained Matrix Factorization problem.

- **User vectors:**
  \[ w_u \in \mathbb{R}^r \]

- **Item vectors:**
  \[ h_i \in \mathbb{R}^r \]

- **Rating prediction:**
  \[ v_{ui} = w_u^T h_i \]

- **Set of non-missing entries**
  \[ \mathcal{Z} = \{(u, i) : v_{ui} \text{ is observed}\} \]

- **Objective:**
  \[
  \arg\min_{\mathbf{w}, \mathbf{h}} \sum_{(u, i) \in \mathcal{Z}} (v_{ui} - w_u^T h_i)^2
  \]
Matrix Factorization (with matrices)

- User vectors:
  \[(W_{u*})^T \in \mathbb{R}^r\]
- Item vectors:
  \[H_{*i} \in \mathbb{R}^r\]
- Rating prediction:
  \[V_{ui} = W_{u*}H_{*i} = [WH]_{ui}\]
Matrix Factorization (with vectors)

- User vectors:
  \[ \mathbf{w}_u \in \mathbb{R}^r \]

- Item vectors:
  \[ \mathbf{h}_i \in \mathbb{R}^r \]

- Rating prediction:
  \[ v_{ui} = \mathbf{w}_u^T \mathbf{h}_i \]

Figures from Koren et al. (2009)
Matrix Factorization (with vectors)

- Set of non-missing entries:
  \[ Z = \{(u, i) : v_{ui} \text{ is observed}\} \]

- Objective:
  \[
  \arg\min_{w, h} \sum_{(u, i) \in Z} (v_{ui} - w_u^T h_i)^2
  \]
Matrix Factorization (with vectors)

- Regularized Objective:

\[
\begin{align*}
\argmin_{w, h} \sum_{(u, i) \in \mathcal{Z}} (v_{ui} - w_u^T h_i)^2 \\
+ \lambda (\sum_i ||w_i||^2 + \sum_u ||h_u||^2)
\end{align*}
\]

Figures from Koren et al. (2009)
Matrix Factorization
(with vectors)

• **Regularized Objective:**

\[
\arg\min_{w,h} \sum_{(u,i) \in \mathcal{Z}} (v_{ui} - w_u^T h_i)^2 \\
+ \lambda \left( \sum_i ||w_i||^2 + \sum_u ||h_u||^2 \right)
\]

• **SGD update for random (u,i):**

\[
e_{ui} \leftarrow v_{ui} - w_u^T h_i \\
w_u \leftarrow w_u + \gamma (e_{ui} h_i - \lambda w_u) \\
h_i \leftarrow h_i + \gamma (e_{ui} w_u - \lambda h_i)
\]

Figures from Koren et al. (2009)
Matrix Factorization  
(with matrices)

• User vectors:  
  \[(W_{u*})^T \in \mathbb{R}^r\]

• Item vectors:  
  \[H_{*i} \in \mathbb{R}^r\]

• Rating prediction:  
  \[V_{ui} = W_{u*} H_{*i} = [WH]_{ui}\]
Matrix Factorization
(with matrices)

- SGD

require that the loss can be written as

\[ L = \sum_{(i,j) \in Z} l(V_{ij}, W_{i*}, H_{*j}) \]

**Algorithm 1 SGD for Matrix Factorization**

**Require:** A training set \( Z \), initial values \( W_0 \) and \( H_0 \)

while not converged do {step}

  Select a training point \((i, j) \in Z\) uniformly at random.

  \[
  W'_{i*} \leftarrow W_{i*} - \epsilon_n \frac{\partial}{\partial W_{i*}} l(V_{ij}, W_{i*}, H_{*j})
  \]

  \[
  H_{*j} \leftarrow H_{*j} - \epsilon_n \frac{\partial}{\partial H_{*j}} l(V_{ij}, W_{i*}, H_{*j})
  \]

  \[
  W_{i*} \leftarrow W'_{i*}
  \]

end while

Figure from Koren et al. (2009)

Figure from Gemulla et al. (2011)